Junior Software Engineer at DevelopmentAid WeatherAPI.com Code Challenge Report

September 3, 2025

Summary:

This document presents a comprehensive overview of the solution developed for the WeatherAPI.com Code Challenge. The application, built in Kotlin using Gradle and Retrofit, successfully retrieves and displays the next day's weather forecast for a list of specified cities. The solution focuses on best practices, API efficiency, and creating a highly maintainable and flexible codebase.

Project Goals & Approach:

The primary objective was to create a command-line application that fetches a one-day forecast for Chisinau, Madrid, Kyiv, and Amsterdam and presents the data in a clear, tabular format.

My approach was guided by three key principles:

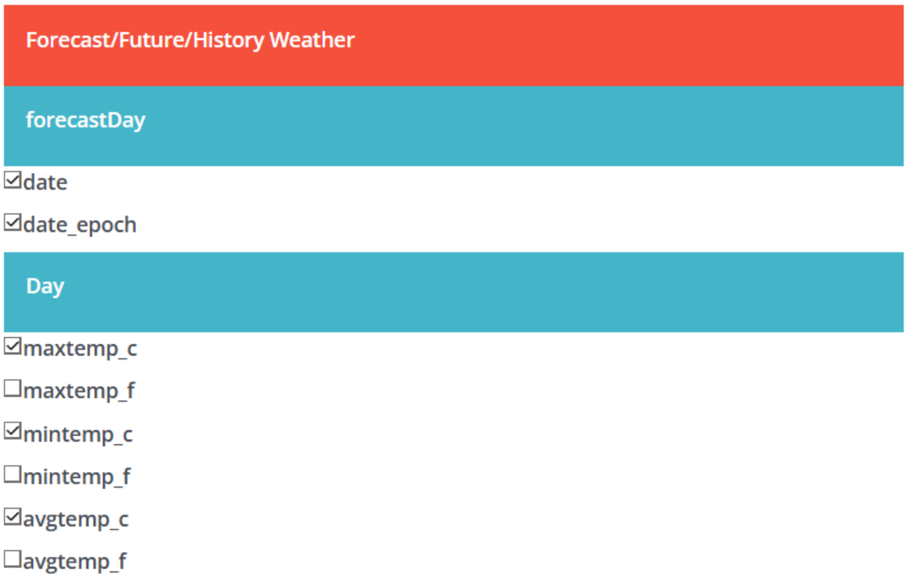
Requirement Analysis: A thorough examination of the API documentation and challenge requirements to ensure all specified data points (Min/Max Temp, Humidity, Wind Speed/Direction) were accurately captured.

Modularity and Best Practices: The use of Kotlin, Gradle, and Retrofit enabled the creation of a clean, well-structured project. This includes dependency management, separation of concerns (API interaction, data processing, and presentation), and modern asynchronous programming with coroutines.

Extensibility and Maintainability: The design allows for easy modification of the target cities and future additions of new features without a significant code refactor.

API Analysis and Data Handling:

The first step was to thoroughly examine the WeatherAPI.com's API documentation, specifically the forecast.json endpoint. A key observation was the large amount of data returned by default, much of which was not required for this challenge (e.g., current conditions, historical data, lunar phases). To optimize API requests and reduce data transfer, I utilized the fields query parameter to disable all non-essential fields, ensuring the application only retrieves what is absolutely necessary for performance and efficiency.


Forecast/Future/History Weather

forecastDay

☑date

☑date_epoch

Day

☑maxtemp_c

☐maxtemp_f

☑mintemp_c

☐mintemp_f

☑avgtemp_c

☐avgtemp_f

Enable required fields only

A notable challenge was the requirement to provide "Wind Direction" for the entire day. The API provides hourly wind direction data but **does not offer a single, day-level metric**. To address this, I implemented a robust solution that provides two distinct calculation methods to choose from.

Method 1 (Max Wind Speed Direction): The direction corresponding to the hour with the highest wind speed throughout the day.

Method 2 (Most Frequent Wind Direction): The most common wind direction observed across all forecast hours.

Project Structure and Technology Stack:

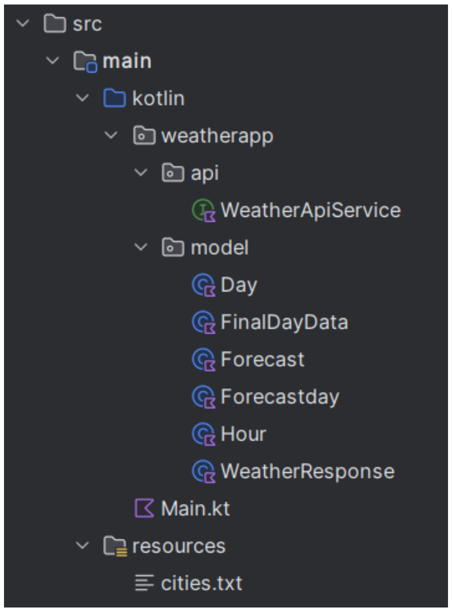
The project is structured to be logical and easy to navigate.

Language: Kotlin was chosen for its conciseness, null safety, and excellent interoperability with Java. Its support for coroutines made handling asynchronous network requests clean and efficient.

Package Manager: Gradle was used to manage dependencies, automate the build process, and package the application.

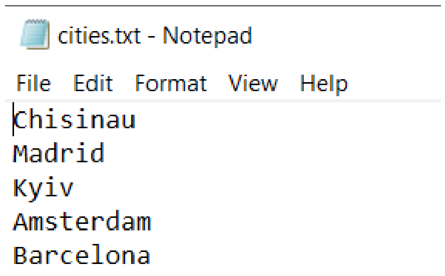
HTTP Client: Retrofit was the ideal choice for API interaction. It simplifies the process of making HTTP requests and parsing JSON responses by converting the WeatherAPI.com JSON data directly into Kotlin data classes, eliminating manual parsing.

Configurability: The list of cities to query is stored in a separate plain-text file (cities.txt). This design decision decouples the business logic from the configuration, making it incredibly simple for a user (or recruiter) to add or remove cities from the list without modifying any source code.



```
src
  main
    kotlin
      weatherapp
        api
          WeatherApiService
        model
          Day
          FinalDayData
          Forecast
          Forecastday
          Hour
          WeatherResponse
        Main.kt
    resources
      cities.txt
```

Project structure



```
Chisinau
Madrid
Kyiv
Amsterdam
Barcelona
```

Configuration file

Git and Version Control:

The project's development was tracked using Git, adhering to a structured branching workflow. All feature development was conducted on dedicated branches, which were then merged into the central dev branch via pull requests. This approach ensured a clean, reviewable, and well-documented history. Each commit message was crafted to clearly describe the changes, providing a transparent and logical history of the development process.

Setup and Execution Instructions:

To set up and run the application, the recruiter should follow these simple steps:

Ensure you have a Java Development Kit (JDK) installed (version 8 or later).

Obtain a free API key from WeatherAPI.com.

Clone the project from the Git repository:

git clone https://github.com/gee-dan/devaid-weather-app.git

Navigate to the project directory and run the following Gradle command (calculate wind direction by highest wind speed over the day):

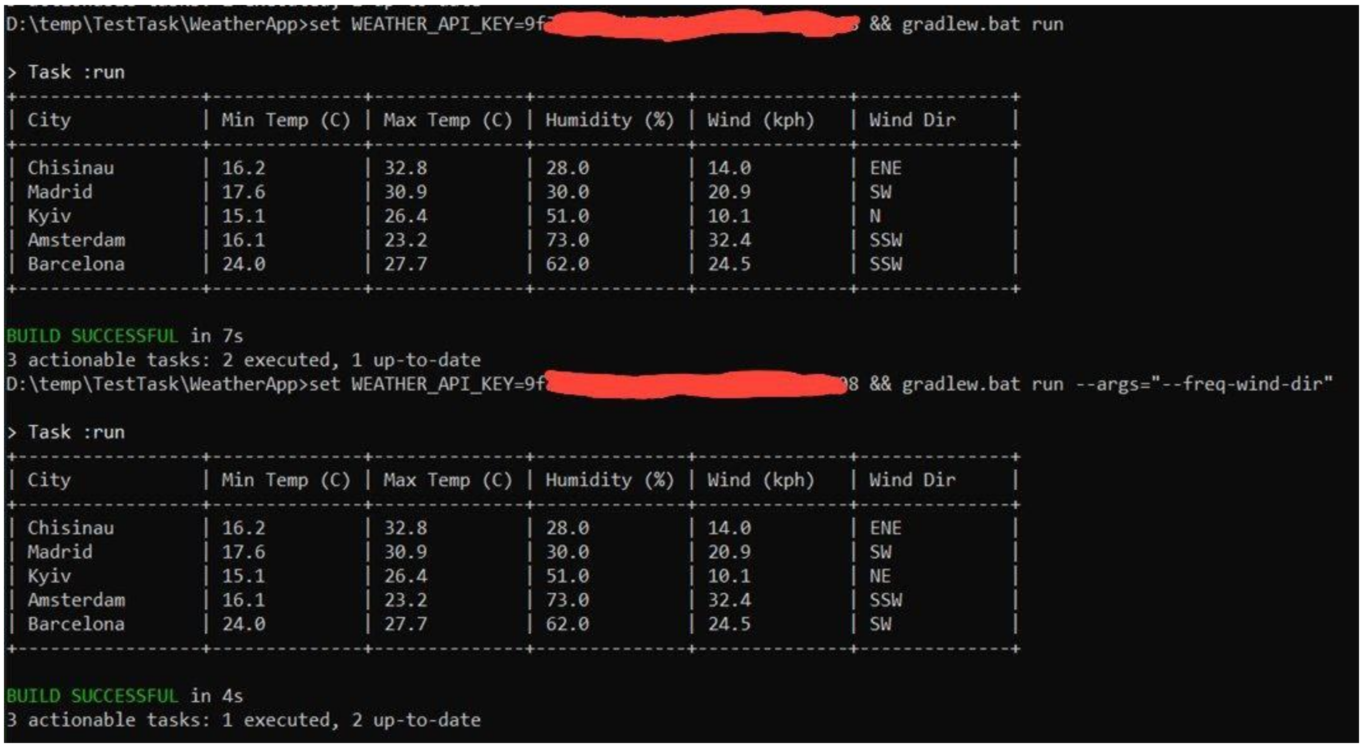
set WEATHER_API_KEY=[YOUR_API_KEY] && gradlew.bat run

or (calculate wind direction by most frequent wind direction over the day):

set WEATHER_API_KEY=[YOUR_API_KEY] && gradlew.bat run --args="--freq-wind-dir"

To get a forecast for different cities, simply edit the cities.txt file in the project's root directory and re-run the application.

Overview:



```
D:\temp\TestTask\WeatherApp>set WEATHER_API_KEY=9f[redacted] && gradlew.bat run

> Task :run
+----------------+--------------+--------------+--------------+--------------+--------------+
| City           | Min Temp (C) | Max Temp (C) | Humidity (%) | Wind (kph)   | Wind Dir     |
+----------------+--------------+--------------+--------------+--------------+--------------+
| Chisinau       | 16.2         | 32.8         | 28.0         | 14.0         | ENE          |
| Madrid         | 17.6         | 30.9         | 30.0         | 20.9         | SW           |
| Kyiv           | 15.1         | 26.4         | 51.0         | 10.1         | N            |
| Amsterdam      | 16.1         | 23.2         | 73.0         | 32.4         | SSW          |
| Barcelona      | 24.0         | 27.7         | 62.0         | 24.5         | SSW          |
+----------------+--------------+--------------+--------------+--------------+--------------+

BUILD SUCCESSFUL in 7s
3 actionable tasks: 2 executed, 1 up-to-date
D:\temp\TestTask\WeatherApp>set WEATHER_API_KEY=9f[redacted]98 && gradlew.bat run --args="--freq-wind-dir"

> Task :run
+----------------+--------------+--------------+--------------+--------------+--------------+
| City           | Min Temp (C) | Max Temp (C) | Humidity (%) | Wind (kph)   | Wind Dir     |
+----------------+--------------+--------------+--------------+--------------+--------------+
| Chisinau       | 16.2         | 32.8         | 28.0         | 14.0         | ENE          |
| Madrid         | 17.6         | 30.9         | 30.0         | 20.9         | SW           |
| Kyiv           | 15.1         | 26.4         | 51.0         | 10.1         | NE           |
| Amsterdam      | 16.1         | 23.2         | 73.0         | 32.4         | SSW          |
| Barcelona      | 24.0         | 27.7         | 62.0         | 24.5         | SW           |
+----------------+--------------+--------------+--------------+--------------+--------------+

BUILD SUCCESSFUL in 4s
3 actionable tasks: 1 executed, 2 up-to-date
```